IN THE SPECIFICATION

Please replace the paragraph starting at line 20 on page 17 through line 27 on page 17 with the following:

Figs. 9A-9E~~8A-8E~~ are an example of a data set 22 being processed via the infrastructure monitor of Fig. 3. Referring to Figs. 9A-9E~~8A-8E~~ and 4, the set of data entries 32 includes 10 rows 64 of entries 62-1..62-10. Four fields A-D are shown in columns 66 for each entry 64. In this exemplary query scenario, each screen 70 displays 3 rows. The user 26 enters an output criteria 42 query requesting screen two, sorted by field C, in descending order. Accordingly, the parser computes that screen two, with a total number of 10 entries 62 and three entries per screen, entries 4-6 define the range 68 of the ordered output display set 40.

Please replace the paragraph starting at line 28 on page 17 through line 5 on page 18 with the following:

Fig. 9B~~8B~~ shows the candidate entries 36 following the first pass 34 and processing by build set handler 46. In this example, the user 36 did not request filtering, hence the filter handler 48 remains idle. The candidate set 36 includes references to the entries 62-4, 62-6, and 62-7 in the set of data entries 32. The candidate set 36 includes the sorting value field C and the index field A, and includes references to the range 68 for the second screen (set) of entries for display (i.e. 4-6) in a descending sort on field C. Therefore, the first pass 34 identifies the entries 62 falling into the range 68 for the particular screen 70 based on a sort of the entire set of data entries 32 on the sort field C. The identified entries 62 need only include an index A and sort value C for the first pass.

Please replace the paragraph starting at line 19 on page 18 through line 9 on page 19 with the following:

In the particular implementation described above, the content data is expressible in an XML format for processing by the SAX parser. The SAX parser is operable to recognize parsing events such as the end of an entry 62, and trigger the corresponding content handler 46, 48, 50, 52,54 component for the event. Therefore, the content handler components are executable entities implementing the operations described above. The SAX parser is responsive to the SAX API such that the XML data triggers parsing events consistent with the normalized structure represented in the XML data. Further, the content handlers may be conformant to the SAX parser such that the handlers implement the interface specified by the SAX parser. Also, the SAX compliant handlers may interoperate with XSL, as is known to those of skill in the art, which is operable to implement transformations on XML data. Further, a Transformation API for XML (TRAX), which defines interfaces and abstract classes, operates in conjunction with SAX and implements the callback invocation discussed above, and further provides a pipelining invocation format. The pipelining coordinates and serializes the content handler 46, 46, 50, 52, 54 invocation and operation such that the handlers and operations may be expressed sequentially, as in the description above, however the actual implementation occurs in an integrated parallel format which preserves atomicity of synchronous events. Accordingly, the methods and operations discussed above in a logical, sequential progression may be implemented by the XSL callback pipelining mechanism in a parallel manner. Alternatively, other implementations provide large dataset processing via sequential and/or~~and./or~~ stack-based function invocation without pipelining.